计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(9):98−103 [doi: 10.15888/j.cnki.csa.008091]
©中国科学院软件研究所版权所有.
E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于国产操作系统独立 GUI 应用研究①

赵正旭[1], 徐　棚[2], 张庆海[1]

[1](青岛理工大学 机械与汽车工程学院, 青岛 266525)
[2](石家庄铁道大学 信息科学与技术学院, 石家庄 050043)
通讯作者: 赵正旭, E-mail: zhaozhengxu@qut.edu.cn

摘　要: 国产 Linux 操作系统运行第三方 GUI 应用软件需要解决软件依赖库问题, 官方提供的依赖软件无法满足依赖库环境配置, 导致大量第三方 GUI 应用软件无法在国产操作系统中安装使用. 现提出一种利用容器技术把第三方 GUI 应用软件及其运行环境打包成独立应用软件的方案, 使第三方 GUI 应用软件能够在国产操作系统上运行. 以开源的分布式渲染系统 Equalizer 为目标对象, 使用 docker 容器技术将其编译环境和运行环境所需的依赖库打包成镜像, docker 镜像在国产操作系统 NeoKylin 上创建容器时配置容器与主机共享 Linux 系统中的 X11 服务, 容器中 Equalizer 解析操作系统中 X11 文件, 在主机屏幕展示图形界面. 本文利用现有的 docker 技术制作独立镜像, 并配置容器与主机系统共享 Linux 系统图形界面服务和显卡驱动程序, 最终实现 Equalizer 程序在国产操作系统环境中正常使用. 实验结果表明, 该方案是可行的, 并可以推广到其他 GUI 应用软件.

关键词: Linux; 国产操作系统; docker; X11

引用格式: 赵正旭,徐棚,张庆海.基于国产操作系统独立 GUI 应用研究.计算机系统应用,2021,30(9):98–103. http://www.c-s-a.org.cn/1003-3254/8091.html

Research on Application of Independent GUI Based on Domestic Operating System

ZHAO Zheng-Xu[1], XU Peng[2], ZHANG Qing-Hai[1]

[1](School of Mechanical & Automotive Engineering, Qingdao University of Technology, Qingdao 266525, China)
[2](School of Information Science and Technology, Shijiazhuang Tiedao University, Shijiazhuang 050043, China)

Abstract: The domestic Linux operating system running third-party GUI application software needs to solve the software dependency library problem. The official dependent software cannot meet the configuration of the dependent library environment, resulting in a large amount of third-party GUI application software that cannot be installed and used in the domestic operating system. A solution is proposed to package third-party GUI application software and its operating environment into independent application software with container technology, so that the third-party GUI application software can run on a domestic operating system. With the open source distributed rendering system, Equalizer, as the target object, the docker container technology is used to package the dependent libraries required for its compilation environment and running environment into a mirror. In the X11 service in the container, the Equalizer in the container parses the X11 file in the operating system and displays the graphical interface on the host screen. This study uses the existing docker technology to create an independent image and configures the container to share the graphical interface service and graphics card driver of Linux system with the host system and finally realizes the normal use of the Equalizer program in the domestic operating system. Experimental results show that the scheme is feasible and can be extended to other GUI application software.

Key words: Linux; domestic operating system; docker; X11

① 基金项目: 河北省自然科学基金 (F2018210058)
Foundation item: Natural Science Foundation of Hebei Province (F2018210058)
收稿时间: 2020-12-07; 修改时间: 2021-01-08; 采用时间: 2021-01-29; csa 在线出版时间: 2021-09-02

© 中国科学院软件研究所 http://www.c-s-a.org.cn

基于 Linux 内核的国产操作系统发行版众多, 却没有一款操作系统完全取代微软桌面系统. 根本原因是国产操作系统的软件生态资源匮乏, 软件种类不全面, 数量不充足导致[1]. 为扩大国产操作系统的应用软件资源, 文献 [1–3] 介绍了在国产操作系统 NeoKylin 环境中部署第三方开源软件的传统方法. 但是由于 Linux 操作系统中开源软件依赖问题, 大部分开源软件无法通过传统的软件部署方法安装, 因此软件依赖问题是主要关注点.

开源软件依赖关系的问题来源于操作系统中的函数库文件. 函数库是为实现某种特定功能而编译好的代码或数据, 以文件的形式存在操作系统上, 是操作系统的重要组成部分. Windows 或 Linux 操作系统中有着大量的函数库文件, 函数库被应用程序调用, 应用软件的运行要依赖很多函数库文件[4].

Windows 作为商业系统其基础函数库相对固定, 第三方软件在此环境中开发, 部署过程中不会有问题. Linux 系统库函数文件分为静态库和动态库. 静态库在软件开发过程中会被包含在生成的二进制文件中. 共享库在程序编译时指定, 程序运行时调用, 不被包含在软件的二进制文件中. 而且开源平台上的软件一般依赖第三方函数库开发, 从而形成软件依赖关系. 所以在国产操作系统中部署第三方软件时会出现找不到依赖函数库的问题[5].

随着容器技术的发展, docker 容器技术能够解决软件依赖问题, 但其主要被用来运行字符界面程序. 针对 GUI 应用程序但不仅限于第三方 GUI 应用程序, 本文的主要工作是: (1) 分析 GUI 应用程序并将其部署在 docker 镜像内; (2) 配置 docker 容器与主机共享 Linux 系统图形界面服务, 并根据 GUI 应用程序对硬件资源的要求, 配置容器与主机共享显卡等驱动程序.

1 相关背景

1.1 国产操作系统

在众多国产操作系统中, 以具有代表性的 NeoKylin (中标麒麟) 操作系统为例, 基于 Linux 内核开发, 是 2010 年国家核高基项目重点扶持的基础软件之一. 当前该桌面系统最新版本为 V7.0, 采用 Linux 3.10 内核. 针对 x86 和国产 CPU 平台自主开发, 打造具有安全可靠和自主可控等特性的国产操作系统. NeoKylin 桌面版操作系统独特的界面设计使其操作简单方便, 系统内置满足日常办公的应用软件[6]. 但是官方提供的应用软件资源, 无法满足用户的日常需求. 在 NeoKylin 上安装其他 Linux 发行版的软件, 通常会因为软件依赖问题, 无法安装运行. 应用软件资源匮乏, 严重制约国产操作系统的发展.

1.2 容器技术

容器技术 (Linux container) 中的容器通常被理解为集装箱, 它把应用程序及程序运行的环境封装在容器内部, 实现应用程序在容器中运行[7]. 在容器的发展当中, docker 容器发展较快其技术也相当成熟, 被视为容器的代表[8]. Docker 容器支持应用打包功能, 把应用程序运行需要的依赖程序和库打包在镜像中, 应用程序就能够在 docker 容器中稳定的执行程序. 容器是 docker 运行镜像后产生的实体, 容器被创建后可以执行暂停和销毁等操作. 容器可看作宿主机上的进程, 某一个容器出现宕机问题不会影响其他容器的正常运行. 虽然容器之间相互隔离, 但是无论在同一台宿主机还是在不同宿主机上的容器可以通过 IP 地址或者 link 机制等其他方式实现容器之间的相互通信[9]. Docker 运行架构如图 1 所示.

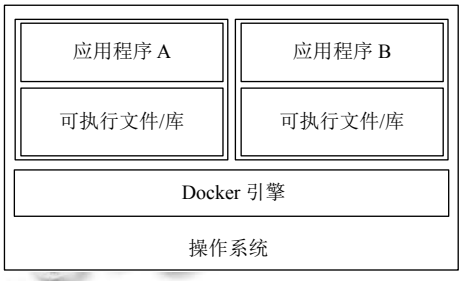

图 1 Docker 容器架构

封装后的镜像可以放在任意 docker 的主机上运行, docker 利用镜像创建容器从而使用容器里的应用程序, 而容器内部应用程序运行完全不依赖主机的运行环境. 封装后的镜像可以放在任意安装 docker 的主机上运行, docker 利用镜像创建容器从而使用容器里的应用程序[10]. 所以对于应用程序, 使用容器技术部署应用软件比传统软件部署方式更有实际意义.

1.3 X11

X11 是 X 协议的第 11 个版本, 也叫做 X Window System, 是一种以位图显示的网络透明化窗口系统[11]. X11 标准是开发可携性图形界面的工业软件标准, 它有一个重要的特性是与设备无关结构, 任何支持 X11 协议的硬件, 就能执行应用程序, 显示图形界面窗口,

© 中国科学院软件研究所 http://www.c-s-a.org.cn

这种特性使得根据 X11 标准开发的应用程序, 可运行在不同环境下 (个人电脑、大型计算机)[12].

X11 是 Linux 图形界面环境的基础, Linux 本质是命令行操作系统, 而 X11 作为应用程序运行在 Linux 操作系统的应用层, 为操作系统提供图形界面显示服务. X11 基于 C/S 架构如图 2 所示, 包括 X server、X client 和 X protocol. X client 是客户端是运行在 Linux 系统上的应用程序, X server 是服务端, 运行在显示设备主机上, X protocol 是客户端和服务端的通信协议.

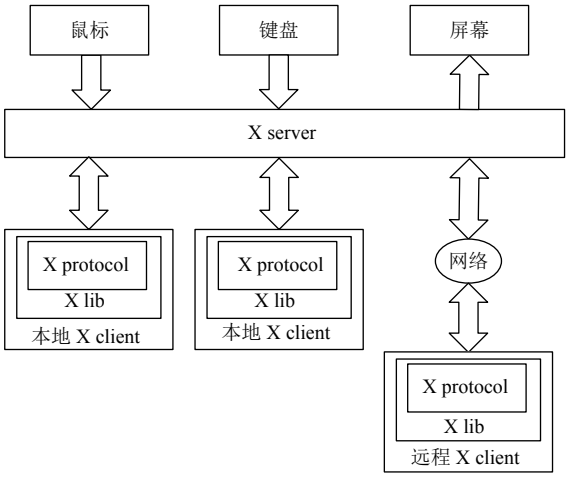

图 2 X11 架构

1.4 Equalizer

Equalizer 一个成熟的并行渲染框架, 它提供了大量的特性、算法和系统集成广泛的现实世界研究和工业应用[13]. Equalizer 提供编译运行方式的 Linux 系统有 Ubuntu 16.04 和 RHEL 6.8[14], 要编译运行 Equalizer 需要安装大量的软件依赖库, NeoKylin 虽然基于 Linux 操作系统, 但官方提供的软件完全不足以编译运行 Equalizer. NeoKylin 使用三方软件, 配置 Equalizer 编译运行环境, 需要进行复杂的工作, 第三方软件还可能与系统自带软件冲突导致系统崩溃. Equalizer作为渲染框架, 在 NeoKylin 操作系统需要 Linux 图形界面服务 X11 的支持, 对软件运行环境的稳定性要求较高, 并且需要使用 GPU 等硬件. 因此以 Equalizer 为目标对象, 进行独立 GUI 应用的实验论证.

2 系统整体方案

2.1 独立应用设计

Docker 镜像是容器运行的实例, 镜像为容器提供文件系统资源. Docker 镜像都是由基础镜像制作而成, 而基础镜像是一般都是 Linux 发行版的 docker 镜像, 例如 Ubunut、Fedora 等.

Linux 操作系统由内核空间和用户空间组成. 内核空间是 kernel, Linux 操作系统启动时会加载 bootfs 文件系统, bootfs 引导 kernel 加载完成后就被卸载掉. Rootfs 是用户空间的文件系统, rootfs 通常包含的是操作系统运行所需的文件系统如/dev、/proc、/bin 等标准目录. 对于不同的 Linux 发行版, 其 bootfs 基本一致, rootfs 不同.

Docker 镜像采用分层的方式构建, 每一个镜像都是由一层一层 union 文件系统叠加组成. Docker 的基础镜像的文件系统分为 bootfs 和 rootfs 两层, 最底端是引导文件系统 bootfs, 在 bootfs 之上的一层是 rootfs. Docker 架构下, 沿用 Linux 中 rootfs 思想. 当 docker daemon 为 docker 容器挂载 rootfs 的时候, 与传统 Linux 内核类似, 将其设定为只读模式. 在 rootfs 挂载完毕后, 与 Linux 内核不同的是, docker daemon 没有将 docker 容器的文件系统设为读写模式, 而是利用 union mount 的技术, 在这个只读的 rootfs 之上再挂载一个读写的文件系统, 挂载时该读写文件系统内空无一物. 可以把 docker 基础镜像创建容器的文件系统理解为: 只含只读的 rootfs 和可读写的文件系统.

容器与主机共享 kernel, 拥有独立的 rootfs, 包含一个操作系统所必须的库文件及软件管理工具, 在容器内部部署软件与主机系统环境无关. 因此 docker 容器能够解决软件依赖问题.

接下来就是制作镜像, 把软件及相关依赖程序打包在镜像内部制作成独立的软件镜像. Docker 制作镜像的方法有两种: 手动制作和 dockerfile 制作. 一般情况下使用 dockerfile 制作镜像, 但是基础镜像只包含操作系统基本函数库文件, 软件需要的一些未知三方库无法通过管理工具安装, 只能根据错误提示查找, 所以先通过手动制作镜像, 再把制作镜像的过程写成 dockerfile 或者 shell 脚本实现自动制作镜像.

2.2 GUI 显示设计

国产操作系统 NeoKylin 采用 X11 提供图形桌面环境, 其 X server 版本号为 1.19.3, X server 在应用层负责接收键盘和鼠标的输入. GUI 应用程序作为 X client, 包含 X lib 和 X protocol. X lib 是 X client 的 C 语言接口库, 封装了 X protocol, 主要提供 X protocol 的存取能力. X server 运行在连接显示屏的主机上, X server 接收并处理来自 X client 的请求, 在屏幕上绘制图形.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

在 NeoKylin 中, X11 通过 DISPLAY 环境变量指定图形显示的屏幕窗口. DISPLAY 的格式是 [unix:端口] 或 [主机名:端口], 前者表示使用本地的 unix 套接字, 后者表示使用 TCP 套接字. X11 服务端默认配置是服务端监听本地的 [unix:0] 端口, DISPLAY 的默认值为 0. 当系统运行 GUI 程序后, 程序会通过本地的 Unix 套接字与 X server 通信, 然后 X server 在当前主机屏幕绘制图形界面.

镜像内部打包的 GUI 应用程序, 在主机屏幕显示图形界面, 需要实现 GUI 应用程序透过容器的隔离机制与主机的 X server 通信. Docker 提供了一种机制, 能够实现把主机上的某个目录与容器的某个目录 (挂载点) 关联起来, 容器内挂载点下的文件就是主机系统目录下的文件, 类似 Linux 操作系统下的 mount 机制. 通过挂载文件的方式, 使主机与容器共享 Unix 套接字, 实现容器内部的应用程序与 X server 通信.

2.3 系统整体架构

Docker 基于基础镜像打包 GUI 软件制作独立的 GUI 应用软件镜像. 软件镜像创建容器, 该容器可以看作独立运行 GUI 应用程序. 实际上 GUI 应用程序是在 docker 容器内运行, 因为容器的隔离特性, GUI 应用程序不能直接与主机上的 X server 通信. 主机上 GUI 程序都是通过套接字实现与 X server 互传信息, 所以利用 docker 文件挂载机制把主机上的套接字文件挂载到容器内部, 实现主机和容器共享套接字. 容器内部的第三方 GUI 应用程序作为 X client, 通过套接字与主机的 X server 通信, X server 把 GUI 应用程序的图形界面绘制在屏幕上. 最终实现 GUI 应用软件在国产操作系统环境中运行. 系统的整体架构如图 3 所示.

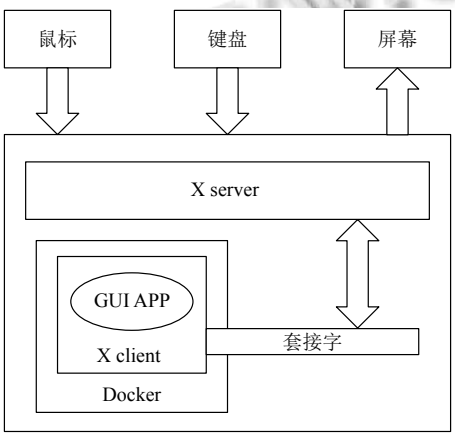

图 3 系统整体架构

3 系统实现

3.1 封装独立 GUI 应用

Equalizer 是开源渲染框架, 在 Github 上公布的源码, 提供了 Linux、Mac OS 和 Windows 三种编译方式. 本文利用 docker 容器做封装工具, 选择 Linux 发行版 Ubuntu16.04 系统镜像, 把 Equalizer 编译并封装在镜像中. 独立的应用程序 (应用程序和程序运行环境) 占用计算机的存储空间, 在制作镜像时要选择占用存储空间较小的镜像. 独立的 GUI 应用程序制作步骤如下:

(1) 以 Ubuntu16.04 的镜像为基础镜像, 使用 docker 创建容器. 在创建容器时注意通过在启动命令中添加--gpus all 参数, 把配置好的 Nvidia 驱动共享给容器.

(2) 进入容器命令行, 按照官方网站给出的 Equalizer 编译环境配置方法操作, 配置 Equalizer 的编译环境.

(3) 下载 Equalizer 源代码, 执行编译命令.

(4) 进入编译后的/bin 文件夹执行示例, 验证结果.

(5) docker commit 命令, 永久保存容器, 生成镜像, 方便下次使用.

Equalizer 封装成镜像之后, 作为独立的 GUI 应用使用. 为方便使用, 镜像创建容器时调用容器内部 shell 脚本, 执行容器内部的应用程序.

3.2 解析 X11 协议

NeoKylin 操作系统, 采用 Xorg 实现 X11 协议, Xorg 提供 X server 服务, 当系统内有程序运行时, 程序会与 X server 通信, X server 接收鼠标、键盘的输入和管理屏幕输出. Docker 内部运行的程序要在屏幕上显示界面, 应用程序就必须要与 X server 通信.

基于国产操作系统上 GUI 应用程序的图形界面展示原理, 采用主机与容器共享 X11 的 Unix 套接字的方式, 实现 X client 的内容从容器内部传递到主机上的 X server. 把主机上的套接字目录挂载到容器内部, 实现主机和容器共享 Unix 套接字.

主机上套接字文件在 tmp 文件夹中, 将其挂载到容器内部, -v /tmp/.X11-unix:/tmp/.X11-unix, 实现主机和容器共享套接字目录. 启动程序后, 底层封装的 Xlib(C 语言接口库) 通过共享 unix 套接字, 传递信息给 X server. X server 代码运行, main 函数进入一个循环. 每次循环都包含, 初始化、处理 X client 信息和退出. 最后 X server 根据 X client 发送的请求, 在主机显示器上绘制图形界面.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

4 实验论证

本章节主要通过具体实验, 验证在国产操作系统环境中方案实施的可行性.

4.1 实验平台

实验平台的计算机软件和硬件配置信息如表 1 所示, 主要包括: CPU、内存、显卡、系统、内核和容器版本信息.

表 1 实验软硬件环境

参数	配置
CPU	(英特尔) Intel(R) Xeon(R) CPU E5-2603 v4 @ 1.70 GHz (1700 MHz)
内存	16 GB
显卡	Nvidia Quadro P2000 (5120 MB)
系统	ReleaseNeoKylin-LinuxDesktoprelease7.0/x86_64-zhaoxinBuild54/20190214
内核	Kernel Linux 3.10.0-862.9.1.nd7.zx.4.x86_64
容器版本	Docker 19.03.13

4.2 实验环境配置

实验环境配置的软件包含: Nvidia 显卡驱动、docker、Nvidia-container-toolkit 工具包.

在 NeoKylin 系统上安装 Linux 版本的 Nvidia 驱动, 驱动要与显卡型号匹配, 否则会安装失败. 在安装之前, 禁用系统默认的驱动, 要把系统运行级别更改为文本模式. 使用 chmod 命令设置驱动程序可执行权限, 再进行安装.

NeoKylin 操作系统上安装 docker 无法直接安装, 直接下载 rpm 包强制安装也无法运行, 因为官方源中没有提供安装 docker 的依赖包. 首先需要给 NeoKylin 操作系统换源, 更换 centos7 的源, 然后引入 docker 源, 直接 yum 下载安装. Linux 系统下, 应用程序稳定运行的基础就是依赖库文件齐全. Yum 安装 docker 组件 docker-ce、docker-ce-cli、containerd.io 时, 操作系统仅提示安装 container-seLinux 依赖和升级 libseccomp 依赖软件. 系统能安装的依赖软件是保存在文件夹的库文件对系统不会有影响, 重点关注升级的依赖. Libseccomp 包为 Linux 内核的系统调用过滤机制提供了一个易于使用且独立于平台的接口, 根据软件向下兼容的特性, 使其版本的升级对系统无影响. 依赖库文件配置齐全, docker 应用在 NeoKylin 系统中即可稳定运行.

Equalizer 是一个渲染框架, 需要用到 Nvidia 显卡, 安装 Nvidia-container-toolkit 工具包, 使 docker 内部能够调用 Nvidia 驱动程序. 安装方法参考 Nvidia-docker 网站提供的安装说明[15], 网站并没有提供针对 NeoKylin 操作系统的工具包, 可使用 CentOS7 的工具包替代.

NeoKylin 操作系统, Xhost 命令是 X server 的访问控制工具. 在当前桌面显示 docker 容器内的 GUI 应用程序, 需要对 X server 设置, 允许 docker 内部的 GUI 程序在显示器上显示. 终端执行 Xhost +, 授予每个用户访问 X server 的权限.

以上是整个实验环境的配置, 完成以上设置后, 使用 docker 制作 Equalizer 渲染框架的镜像.

4.3 制作并运行独立 GUI 应用

NeoKylin 环境中制作 GUI 应用程序镜像并运行步骤是: (1) 基于 Ubuntu16.04 镜像创建容器, 创建命令中配置挂载 X11 的 Unix 套接字, 并引入显卡驱动程序; (2) 在容器内部安装部署 Equalizer; (3) 容器内运行 Equalizer 程序, 测试启动是否正常; (4) 主机内执行命令, Equalizer 镜像创建容器并自动运行 Equalizer.

编写 shell 脚本, 自动配置以上操作. 主要包括以下 4 个脚本:

1) 脚本 Run.sh 在主机上运行, 负责运行 ubuntu16.04 镜像创建容器, 主要内容如下 (#后为注释):

-v /tmp/.X11-unix:/tmp/.X11-unix #设置主机与容器共享套接字

-gpus all #设置容器与主机共享显卡驱动

2) 脚本 Equalizer.sh 在容器内部运行, 负责编译 Equalizer 源码, 主要内容如下:

apt-get install … #安装编译环境

git clone … #下载源码

ninja #编译

3) 脚本 Init.sh 在容器内部运行, 负责启动 Equalizer 可执行文件, 主要内容如下:

cd /Equalizer/build/bin/#进入编译后的文件目录

./eqPly #执行可执行文件

4) 脚本 EqualizerRun.sh 在主机上运行, 负责启动容器内部 GUI 程序, 主要内容如下:

-v /tmp/.X11-unix:/tmp/.X11-unix #设置主机与容器共享套接字

-v /Equalizer:/Equalizer#主机与容器共享文件夹

-e DISPLAY=unix$DISPLAY #设置主机显示屏

-gpus all #共享显卡驱动

Docker 运行 Equalizer 框架镜像, 使用分布式渲染

© 中国科学院软件研究所 http://www.c-s-a.org.cn

时, 容器和主机要设置 IP 和端口号, 因为分布式渲染是通过网络实现通信.

4.4 实验结果分析

在 NeoKylin 操作系统中直接配置 Equalizer 编译安装环境, 由于官方源提供的软件不足以配置 Equalizer 的编译环境, 所以下载相关依赖软件源码编译安装, 而该软件又依赖其他软件, 整个过程陷入一个循环之中. 在安装软件过程中, 因为第三方依赖软件问题多次导致系统无法启动, 未能成功安装编译.

Equalizer 渲染框架源码大小 100.9 MB. 在容器内配置编译环境需要安装的软件及相关依赖库共 394 个, 需要 269 MB 存储空间, 安装完成使用 978 MB 的额外磁盘空间. 源码编译完成后制作的镜像为 2.82 GB, 导出镜像包为 2.68 GB.

为测试容器内 Equalizer 程序的稳定性, 对其进行压力测试. 使用两个节点渲染, 增加渲染模型的大小比较模型加载时间和渲染的帧率. 实验数据如表 2 所示.

表 2 Equalizer 渲染不同模型产生的数据

模型大小	加载时间(s)	帧率 (帧/s)
4.4 MB	0.8	27.2
1.05 GB	359.65	19.9

实验结果表明, 在 NeoKylin 环境中使用 docker 容器运行 Equalizer 渲染的模型越大, 加载时间越久, 帧率降低. 虽然总体渲染效率下降, 在该环境下 Equalizer 不仅能够对 4.4 MB 的小模型进行渲染, 而且能够渲染 1.05 GB 的大模型, 证明其具有稳定性, 也表明在国产操作系统 NeoKylin 环境中, 使用 docker 容器运行 GUI 应用程序的方案是可行的.

5 总结与展望

本文对国产操作系统环境中制作并运行独立 GUI 应用程序进行探索, 利用容器技术初步实现制作独立 GUI 应用程序在国产操作系统上运行. 结果表明在 NeoKylin 国产操作系统上, 使用 docker 容器技术, 可以实现第三方 GUI 应用程序独立运行. 文中方案给国产操作系统兼容第三方 GUI 应用程序提供新思路, 使用该方案可以增加国产操作系统的应用软件资源. 但方案中还有许多细节问题需要研究, 包括容器和主机共享文件的机制, 完善容器技术或可解决国产操作系统软件资源匮乏的问题, 有助于加快国产操作系统的发展.

参考文献

1 赵正旭, 陶智, 徐骞. 基于国产操作系统应用软件部署对策的探讨. 微型机与应用, 2016, 35(18): 16–18. [doi: 10.19358/j.issn.1674-7720.2016.18.004]

2 赵正旭, 白英杰, 吴晓进. 国产操作系统 JSP 服务器部署策略的设计与实现. 软件, 2018, 39(6): 196–200.

3 白英杰, 赵正旭, 吴晓进, 等. 国产操作系统 PHP 服务部署策略的设计与实现. 计算机应用与软件, 2019, 36(1): 17–21. [doi: 10.3969/j.issn.1000-386x.2019.01.004]

4 赵广涛. 开源平台依赖关系解决方案的研究与实现 [硕士学位论文]. 开封: 河南大学, 2012.

5 王静. 基于软件网络模型的开源操作系统复杂性及演化分析技术研究 [博士学位论文]. 长沙: 国防科技大学, 2019. [doi: 10.27052/d.cnki.gzjgu.2019.000023]

6 赵正旭. 麒麟操作系统使用与推广. 北京: 科学出版社, 2014. 1–8.

7 Madhumathi R. The relevance of container monitoring towards container intelligence. Proceedings of 2018 9th International Conference on Computing, Communication and Networking Technologies. Bengaluru, India. 2018. 1–5.

8 郎丰凯. 基于云计算的容器技术概述. 中国新通信, 2019, 21(21): 123–124.

9 陈伟, 叶宏杰, 周家宏, 等. 基于领域知识的 Docker 镜像自动构建方法. 大数据, 2021, 7(1): 64–75.

10 房锦章, 武延军. 基于 Docker 技术的 GUI 应用的在线迁移研究. 计算机系统应用, 2016, 25(10): 246–251. [doi: 10.15888/j.cnki.csa.005351]

11 胡皓. X Window 实务应用. 北京: 人民邮电出版社, 2000. 27–33.

12 江帆, 贺也平, 周启明. SurfaceFlinger 在 X Window 系统环境下的运行方案. 计算机系统应用, 2017, 26(10): 95–101. [doi: 10.15888/j.cnki.csa.006012]

13 Eilemann S, Steiner D, Pajarola R. Equalizer 2.0 — Convergence of a parallel rendering framework.. IEEE Transactions on Visualization and Computer Graphics, 2020, 26(2): 1292–1307. [doi: 10.1109/TVCG.2018.2870822]

14 Eyescale/Equalizer. Equalizer. https://github.com/Eyescale/Equalizer. (2019-11-14).

15 Nvidia. Nvidia-docker. https://nvidia.github.io/nvidia-docker. (2019-09-16).

© 中国科学院软件研究所 http://www.c-s-a.org.cn